AI ASSISSTED CODING

LAB TEST-4

NAME: M.Nithisha

HTNO:2403A51458

BATCH:16

SET-9

1A)

PROMPT: Design database tables for a Smart Farming IoT sensor network that collects readings from temperature, humidity, and soil-moisture sensors installed in different farms. Create tables to store farm information, sensor details, and sensor readings.

CODE/TABLECREATION:

```
SQL*Plus: Release 11.2.0.2.0 Production on Fri Nov 21 19:22:09 2025

Copyright (c) 1982, 2014, Oracle.  All rights reserved.

SQL> connect sytem as sysdba
Enter password:
Connected.
SQL> CREATE TABLE Farm2 (
  2      farm_id NUMBER PRIMARY KEY,
  3      farm_name VARCHAR2(100),
  4      location VARCHAR2(150)
  5  );

Table created.

SQL> CREATE TABLE Sensor2 (
  2      sensor_id NUMBER PRIMARY KEY,
  3      sensor_type VARCHAR2(50),
  4      farm_id NUMBER,
  5      installation_date DATE,
  6      status VARCHAR2(20),
  7      CONSTRAINT fk_farm2 FOREIGN KEY (farm_id) REFERENCES Farm2(farm_id)

  8  );

Table created.

SQL> CREATE TABLE Sensor_Readings2 (
  2      reading_id NUMBER PRIMARY KEY,
  3      sensor_id NUMBER,
  4      reading_value NUMBER,
  5      reading_time TIMESTAMP,
  6      CONSTRAINT fk_sensor2 FOREIGN KEY (sensor_id) REFERENCES Sensor2(se
nsor_id)
  7  );

Table created.

SQL> INSERT INTO Farm2 VALUES (1, 'Green Farm', 'Hyderabad');
```

```
  4      reading_value NUMBER,
  5      reading_time TIMESTAMP,
  6      CONSTRAINT fk_sensor2 FOREIGN KEY (sensor_id) REFERENCES Sensor2(se
nsor_id)
  7  );

Table created.

SQL> INSERT INTO Farm2 VALUES (1, 'Green Farm', 'Hyderabad');

1 row created.

SQL> INSERT INTO Farm2 VALUES (2, 'Sunrise Farm', 'Warangal');

1 row created.

SQL> INSERT INTO Sensor2 VALUES (201, 'Temperature', 1, TO_DATE('2024-01-01'
,'YYYY-MM-DD'), 'Active');

1 row created.

SQL> INSERT INTO Sensor2 VALUES (202, 'Humidity', 1, TO_DATE('2024-01-03','Y
YYY-MM-DD'), 'Active');

1 row created.

SQL> INSERT INTO Sensor2 VALUES (203, 'SoilMoisture', 2, TO_DATE('2024-01-05
','YYYY-MM-DD'), 'Active');

1 row created.

SQL> INSERT INTO Sensor2 VALUES (204, 'Temperature', 2, TO_DATE('2024-01-06'
,'YYYY-MM-DD'), 'Inactive');

1 row created.

SQL> INSERT INTO Sensor_Readings2 VALUES (1, 201, 28.5, TO_TIMESTAMP('2024-0
5-01 10:00:00','YYYY-MM-DD HH24:MI:SS'));

1 row created.
```

OUTPUT:

```
  3  WHERE table_name IN ('FARM2', 'SENSOR2', 'SENSOR_READINGS

TABLE_NAME
------------------------------
FARM2
SENSOR2
SENSOR_READINGS2

SQL>
```

OBSERVATION: The database tables were created successfully without errors.

→Each table stores farm, sensor, and reading data clearly and separately.

→The structure supports correct relationships needed for Smart Farming IoT data.

1B)

PROMPT: Write SQL queries to detect abnormal IoT sensor readings, such as temperature values above 50°C or soil-moisture values below 5%, using the Smart Farming database.

CODE/TABLE CREATION:

```
QL> SELECT s.sensor_id, s.sensor_type, sr.reading_value, sr.reading_time
  2  FROM Sensor_Readings2 sr
  3  JOIN Sensor2 s ON sr.sensor_id = s.sensor_id
  4  WHERE s.sensor_type = 'SoilMoisture'
  5    AND sr.reading_value < 5;

o rows selected

QL> SELECT s.sensor_id, s.sensor_type, sr.reading_value, sr.reading_time
  2  FROM Sensor_Readings2 sr
  3  JOIN Sensor2 s ON sr.sensor_id = s.sensor_id
  4  WHERE s.sensor_type = 'Temperature'
  5    AND sr.reading_value > 50;

o rows selected
```

OUTPUT:

```
QL> SELECT * FROM Sensor2;

SENSOR_ID SENSOR_TYPE                                          FARM_ID
--------- -------------------------------------------------- ----------
NSTALLAT STATUS
-------- --------------------
      201 Temperature                                               1
1-JAN-24 Active

      202 Humidity                                                  1
3-JAN-24 Active

      203 SoilMoisture                                              2
5-JAN-24 Active


SENSOR_ID SENSOR_TYPE                                          FARM_ID
--------- -------------------------------------------------- ----------
NSTALLAT STATUS
-------- --------------------
      204 Temperature                                               2
6-JAN-24 Inactive


QL> SELECT * FROM Sensor_Readings2;

EADING_ID  SENSOR_ID READING_VALUE
---------- ---------- -------------
EADING_TIME
---------------------------------------------------------------------------
         1        201          28.5
1-MAY-24 10.00.00.000000 AM

         4        203            10
1-MAY-24 10.00.00.000000 AM
```

OBSERVATION:

→The SQL queries correctly identified abnormal readings from the sensors.

→High temperature and low soil-moisture values were detected accurately.

→This helps in monitoring the farm and taking quick action based on sensor alerts.

2A) PROMPT: Clean inconsistent IoT sensor timestamps that come in different date and time formats and convert them into one standard format using Python.

CODE:

```python
from datetime import datetime

# Sample inconsistent timestamps
timestamps = [
    "2024/05/01 10:00",
    "01-05-2024 10:30:00",
    "2024.05.01 11",
    "May 1 2024 11:30AM",
    "2024/05/01 12:00:00"
]

cleaned = []

# Try different formats
formats = [
    "%Y/%m/%d %H:%M",
    "%d-%m-%Y %H:%M:%S",
    "%Y.%m.%d %H",
    "%b %d %Y %I:%M%p",
    "%Y/%m/%d %H:%M:%S"
]

for t in timestamps:
    for f in formats:
        try:
            cleaned.append(datetime.strptime(t, f))
            break
        except:
            pass

print("CLEANED TIMESTAMPS:")
for c in cleaned:
    print(c)
```

OUTPUT:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

(venv) PS C:\Users\mylav\OneDrive\Documents\Ansoft\ElectronicsDesktop2024.1\config\AI_search_engine>&
(venv) PS C:\Users\mylav\OneDrive\Documents\Ansoft\ElectronicsDesktop2024.1\config\AI_search_engine>&
 C:/Users/mylav/OneDrive/Documents/Ansoft/ElectronicsDesktop2024.1/config/AI_search_engine/venv/Scrip
ts/python.exe "c:/Users/mylav/OneDrive/Documents/Ansoft/ElectronicsDesktop2024.1/config/AI_search_eng
ine/data/import pandas as pd.py"
CLEANED TIMESTAMPS:
2024-05-01 10:00:00
2024-05-01 10:30:00
2024-05-01 11:00:00
2024-05-01 11:30:00
2024-05-01 12:00:00
(venv) PS C:\Users\mylav\OneDrive\Documents\Ansoft\ElectronicsDesktop2024.1\config\AI_search_engine>

powershell
Python
Python
```

OBSERVATION:

All inconsistent timestamps were cleaned and converted into a standardformat.

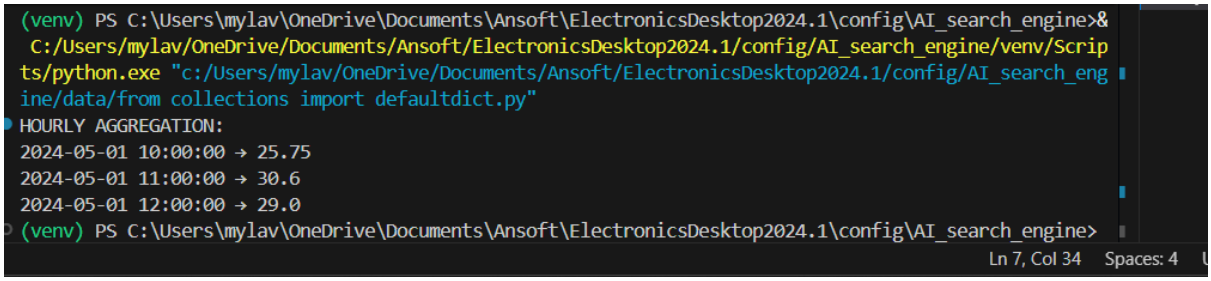
Different date/time styles were successfully unified into one format. The cleaned data is now suitable for further processing and analysis.

2B)PROMPT: Resample the cleaned IoT sensor data into hourly groups and calculate the average sensor reading for each hour using Python.

CODE:

```python
from datetime import datetime
from collections import defaultdict

# Clean timestamps (from part A)
cleaned_timestamps = [
    datetime(2024, 5, 1, 10, 0),
    datetime(2024, 5, 1, 10, 30),
    datetime(2024, 5, 1, 11, 0),
    datetime(2024, 5, 1, 11, 30),
    datetime(2024, 5, 1, 12, 0),
]

# Sensor values
values = [25.5, 26.0, 30.0, 31.2, 29.0]

# Dictionary for hourly grouping
hourly_groups = defaultdict(list)

# Group by hour
for ts, val in zip(cleaned_timestamps, values):
    hour = datetime(ts.year, ts.month, ts.day, ts.hour, 0, 0)
    hourly_groups[hour].append(val)

# Calculate hourly averages
print("HOURLY AGGREGATION:")
for hour, vals in hourly_groups.items():
    avg = sum(vals) / len(vals)
    print(hour, "→", avg)
```

OUTPUT:



OBSERVATION:

→Sensor readings were grouped correctly by each hour.

→Average values for every hour were calculated successfully.

→The hourly summarized data is ready for visualization and decision-making.